Risk Assessment and Mitigation

"Mathochist Studios" Cohort 4, Team 11

Euan Cottam
Charlie Thoo-Tinsley
Harri Thorman
Will King
Zach Moussallati
Aiden Turner
Marcus Williamson
Joshua Zacek

As part of software engineering , there are a number of threats that can potentially hinder the development, and thereby the performance, of a system. Moreover, it's important to recognise that these risks can appear at any point during development. Thus it's of equal importance to define procedures for managing these aforementioned risks before they arise.

Our risk management strategy can be divided into 4 components: identification, analysis, planning and monitoring.
1) Firstly, identification is the discovery of plausible risks to the project (excluding those with a low probability or of minor consequence). Risk identification should utilise knowledge from all members of the team and each identified risk should be documented in the risk register. This stage is essential as you can't manage risks without first observing which ones are present. Each risk should be designated a risk ID, a type and a description.
2) The second stage of the risk management process is analysis. Here each of the identified risks are assigned ratings for likelihood and severity. This approach allows us to prioritise risks which pose a greater threat to the project, further mitigating the overall risk potential of the project.
3) Risk planning involves creating mitigation and avoidance strategies for the individual risks we have identified. Avoidance strategies highlight the things that should be avoided that would otherwise cause risk to the project, reducing the likelihood of risks to occur. Similarly the mitigation strategies reduce the severity, or impact, of each risk. 4) Team members are assigned a risk (ownership) and are responsible for monitoring their risks. This includes regularly reviewing the assigned likelihood and severity ratings as well as reporting on the status of risks. As this is a small group project, it's enough for the reporting to be done verbally at either of the weekly group meetings.

We're using a risk register as a comprehensive document of all of the risks we have identified. Each risk has an ID, type, description (identification) as well as their assessed likelihood and severity (assessment). There is also a section to describe how we're going to mitigate each risk. Lastly each risk is assigned an owner who is responsible for monitoring it and communicating its status to the relevant stakeholders:

1. Risk ID - A numerical ID given to each risk which helps to track risks throughout the project.
2. Type - What type of risk it is. It is split into three distinct categories: Business, Product and Project.
3. Description - A short write up of what each risk is about.
4. Likelihood - The chances of one of these risks happening or recurring throughout the project.
5. Severity - How much of an impact can be done each risk will make to the project.
6. Mitigation - Solutions that can be done to either prevent the risk happening again or minimise the effect.
7. Owner - Who will be responsible for each risk.

Risk ID	Type	Description	Likelihood	Severity	Mitigation	Owner
R1	Product and Project	Version control conflicts when multiple members edit at the same time	L	M	Effective communication amongst the group to prevent conflicting versions	Aiden
R2	Project	Members absent for various reasons	H	H	Reassignment of tasks and roles + work to be done on a shared drive to track progress	Euan
R3	Project	Underestimating time required for key and core game mechanics	H	H	Setting weekly specific goals, progress is discussed at meetings	Josh
R4	Project	Assets were all made by the group not sourced which took a while	H	L	Will took the lead and his other initial workload was disseminated	Will
R5	Project	JAR does not run consistently across Windows / Mac / Linux	L	H	Avoided OS specific dependencies	Aiden
R6	Business	Misinterpreting client expectations for events/difficulty amongst the group	L	M	Documented the client meeting and clarified with each other	Charlie
R7	Project	Poor team communication resulting in missed meetings and unclear tasks	M	M	Bi-weekly set meetings + table showing what everyone's tasks and roles are	Josh

R8	Product	Gameplay bugs affect core mechanics	H	M	Manual weekly testing done by all of the group	Marcus
R9	Product	Low code quality / inconsistent style	M	M	Coding style standardised and reviewed weekly	Marcus
R10	Product	Lack of early user feedback before handover	L	M	Early testing amongst friends and peers, feedback documented	Charlie
R11	Project	Risks were not identified at the start of the project	M	M	Risks recorded and identified throughout the project	Harri
R12	Product	Lack of prior experience with LibGDX	H	L	Those with experience took the lead	Aiden
R13	Product	Performance issues across different hardware	H	L	Tested across multiple different hadrwares	Marcus
R14	Project	Documentation inconsistent with the code (Not updated UML Diagrams)	M	H	Documentation done in tandem with implementation	Zach
R15	Project	Game files lost due to corruption	L	H	Regular copies of the game saved	Aiden

R16	Project	Members falling behind due to prioritising other academic workload / other commitments	H	L	Good communication and workload shared accordingly	Josh
R17	Product	Dependencies and 3rd party libraries become incompatible / unavailable	M	H	Alternative options considered and downloading copies of the dependencies	Aiden
R18	Product	Game is either too easy or too hard to complete ruining the player's experience of the game	M	H	Continually testing and playing the game during development and after adding certain events / components that could change the difficulty.	Marcus